Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 6– No.10, April 2017 – www.caeaccess.org

A New Machine Learning based Approach for Text Spam Filtering Technique

Dipmalya Sen
Department of CSE
B.P.P.I.M.T, Kolkata, India

Chandan Das
Department of AEIE
BCREC, Durgapur, India

Sarit Chakraborty
Department of CSE
B.P.P.I.M.T
Member, IEEE, Kolkata, India

ABSTRACT
Electronic mail (e-mail) has become an essential element in our daily activities in recent past. Volume of email traffic is increasing many a fold in last couple of decades. Out of all such e-mails around 80% are unwanted mails, called as unsolicited bulk email (UBE) or spam mails. With the drastic increase in the use of electronic mail, there has also been an escalation in the problem of dealing with spam mails. In spite of availability of many commercial text based spam filters, users still suffer from the problem of spam mail, which unnecessarily accumulated in their inbox.

In this work, we have proposed a spam detection algorithm based on Machine Learning approach. We have used the concept of Cumulative Weighted Sum (CWS) seeking to achieve a greater rate of accuracy in detecting spam mails. Three different techniques are also proposed for calculating CWS value. Our method is able to detect most of the spam and provides an accurate and dynamic filtration for such mails. Experimental results of our technique with different benchmark datasets are quite significant and gives much improved performance than the available text spam filters.

Keywords
E-mail, Spam, Ham, Machine learning, Naïve-Bayes, Cumulative-Weighted Sum

1. INTRODUCTION
The most effective and formal method of communication in current days is Electronic mail commonly known as 'E mail'. More than 500 million people in the world have internet access and the popularity of email technology has grown rapidly in recent years [7]. But, the day to day increase in the number of spam mails has caused a big reason of dissatisfaction amongst the users. Spam mails not only hamper the user's mailing experience but also in many cases become the source of computer virus or malware as well and negative impact on the user's professional as well as personal life.

Until recent past, this problem was only stuck to text-based spam mail. Presently spammers have taken a new approach; where apart from sending the spams by text form they send it via image files like .jpg, .png, or .gif formats [5-6]. To prevent these spammers and to give the users a better mailing experience, a number of methodology have been proposed till date [1-3], which include algorithms like Support Vector Machines, Naive-Bayesian [10-11], Decision tree classifiers based methods [7-9] and other machine learning techniques [4]. Many spam filtering software have also been developed over the years. But, still the problem of spam mail is present as it was. In most of the cases, a few organizations take this spam mail as an important tool for advertising. Mostly, they send fake links showing a lust for offers or prizes and draw visitors, and in other occasions, it is mostly pranks.

There are many solutions to spam filtering, e.g., the blacklist and white-list filtering techniques [14], decision tree based approaches [7], [8], [9] and machine learning based methods [4], [15]. Among various solutions, machine learning based ones are receiving more attention due to its high accuracy rate for spam detection.

For detecting text spam mails, initially we have chosen a trivial list of sample keywords with a pre-assigned weightage for each key. As soon as, the user allows access to the application, the proposed machine-learning algorithm will start scanning different parameters at the client site and analysed the available data like the user inbox, sent items, contacts and alongside the browsing history etc. Based on these parameters the data analysed the keywords list will get updated. This procedure will be then followed by fixation of weightage to each mail based on the algorithm of Cumulative Weighted Sum and check whether the weightage crosses the granted threshold limit. If it does, then the mail can be considered as spam else non-spam (ham).

A complete client-based spam-detection method is proposed in this work. The model when implemented as application software can be attached as a plug-in to any browser. By allowing the application to access one's e-mail, this will automatically detect spam mails and give the user a better mailing experience with customizable user needs.

2. PRELIMINERIES
Spam mails or unsolicited mails can be categorized into two major types- i.) Content or text-based spam mail [12-13] and ii.) Image based spam mails [5-6].

2.1 Text Spam mails
The most general types of mails that are sent across the internet are mainly in the text format. These include mostly advertisements and offers in text format. This format of spam mails is sent through mostly as SMS and E-mails.

2.2 Image Spam mails
In recent times, spammers have adopted this new technique, so that the mails can go undetected through any text spam filters. Advertisements, Offers, Lotteries and other mails that were been sent via text in the past days are now being sent in the form of images. Image spam exploded in 2006 and by early 2007 it had reached a peak of over 50% of total spam received and its menace is still going. Some typical image-based spam are shown in Fig. 1

Fig. 1: **A typical example of image-based spam mail**

3. METHODOLOGY

The proposed method in this paper for spam mail detection is based on Machine Learning concepts. The focus of our work is given for superior user experience and customizable spam-detection mechanism according to the specific needs of a user. The detailed analysis of our method named as User's-based Machine Learning Algorithm (UMLA) is as follows:

3.1. User-based Machine Learning Algorithm

For detecting a text-based or image-based spam mail, the primary criterion is the content of the mails we are dealing with. Then to detect whether the content is good or bad, legitimate or illegal, having some perceived value or useless based on the user's point of view has been taken into consideration. We have to tally whether the keywords in the mail's content match with the set of keywords predefined in our algorithm's list. This set of keywords are not static in nature, rather it depends on six different parameters as below:

- The user's choice of words in his own sent-mails
- The contact list of the user's email account
- The user's browsing history
- The accounts opened from the browser
- The user's inbox and drafts
- User's profile e.g. Age, Sex, Ethnicity, Locations of living etc.

From the six-tuple analysis as above, the algorithm will update the pre-assigned keyword list and re-assign the keywords along with their weightage. Let us now, elaborate the six-tuple based method of analysis:

1. The choice / uses of words vary from user to user. The word that may be irrelevant to a particular user might have some relevance to some other user. So, to get a complete overview of the user's choice, the algorithm primarily checks the sent mails of the users. After scanning the sent mails, if the keyword counter detects the abundance of a particular keyword which was primarily present in the default keyword list, then there will be two choices, firstly, to reduce the weightage of that particular keyword or secondly, to remove the keyword from the keyword list.

For example, let us assume a user, who is a property dealer / developer. Primarily, the keyword list had words like "Offers", "Sale", "Discount", etc. as blacklisted words. But for this user, from analysis, it was seen that the words "Discount" and "Sale" are being used by the user himself in many occasions. So, it as per the algorithm, depending on the number of occurrences of these keywords, the spam-weightage of these words will either be diminished or removed from the black-list. This process will continue periodically and if it is seen that the user has decreased or ceased the use of such words, then depending on the current usage, those keywords can again be added to the spam-list. Thus the list gets re-checked and re-formed in every 30 days' time period dynamically. This dynamicity of the entire process is one of the significant feature of our algorithm proposed and gives much accurate spam detection according to user's need.

2. The next most important analysis before fixing weight to any mail is analyzing the contact list of the user. If a mail is being sent from someone who belongs to the contact list of the user, then it is assumed that the sender's mail is non-spam and it is a trusted source. Hence, this analysis would reduce the effort of scanning the mail.

3. To detect whether a mail is spam or not, another necessary step is to check the user's browsing history. This is where the Browser Intelligence System (BIS) comes into picture. Our proposed algorithm will go through the browser's history, bookmarks and the most visited websites for last three months. If from analysis, it is observed that some mail is coming from a domain, which the user visits on a regular basis, then the algorithm will put that domain or website into the trusted list and the mail from that website will be treated as non-spam.

For example, let us assume a user visits www.xyz.com on almost a regular basis ($\geq$ some threshold value Υ and time duration taken as 30 days), and that website sends the user mails with domain xyz.com. Under this scenario, the mail will not be treated as a spam mail by default. But, since the algorithm is dynamic, if the user stops visiting that website for a long time and if mails from that website remain unseen, then those mails will go to spam mail folder if those are spam indeed considering other criteria.

4. The next application of the Browser Intelligence System is done by checking all the accounts that are opened from that browser or the user uses the same email id for other websites. If this happens, then the algorithm will put that domain or website into the trusted list, and all the mails from that website will be treated as legitimate mails.
For example, let us assume the user uses his email id for his account in some E-Commerce website and is a user of that website. Then even if any notice regarding sale or discount comes from that website, then that mail will be treated as a useful mail for the user.

5. The next analysis is the analysis of the user's Inbox as well as mails in Drafts. If the scenario happens such that the user visits the Spam mail folder and checks any mail and click any link in that mail or save it as Draft, then that source will be treated as a trusted source.

6. In the last analysis, the algorithm would check the user's profile, his/her age and gender, ethnicity, locations of living, social background and depending on that there would be a final filter on the keyword list and also the weight fixation for the keywords will be done accordingly. The pseudo-code of our method (UMLA) is given in Fig. 2.

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 6– No.10, April 2017 – www.caeaccess.org

3.2 Cumulative Weight Fixation:

Once our User-based Machine Learning Algorithm (UMLA) is applied, then the successive steps that will keep on following for every mail is affixing weight to each mail. The keyword list generated from the UMLA will be used to assign weightage to each mail. If the weightage exceeds the pre-defined threshold value then the mail will be treated as a spam. To make this process efficient and quick, we have divided the task into two parts: a. Searching Algorithm adopted for keyword listing and b. Weight Fixation for each mails.

Inputs:

i. The set of pre-assigned keywords $k_1, k_2, k_3 \ldots k_n \in K$
ii. The set of weightage values for the respective keywords $w_1, w_2, w_3, \ldots w_n \in W$
iii. Threshold value: W_t , Contact list : C_L

Pre-Process:

1. Check Sent Mails
   1.1. If(count(k_i) > W_t)
   then, drop(k_i) and drop(w_i) from set K and W respectively; where $k_i \in K$ and $w_i \in W$
2. Check Contact
   2.1 if(mail →sender ∈ C_L)
   Treat the mail as non-spam
3. Check Browsing History of the Browser
   3.1 if(mail→sender ∈ website_visited)
   Treat the mail as non-spam
4. Check Accounts saved in Browser's cache
   4.1 if(mail→sender ∈ website_in_cache)
   Treat the mail as non-spam
5. Check User's Profile
   5.1 if(user's age < 18)
   Drop or Add certain keywords
   5.2 if(user's gender = = 'Female')
   Drop or Add certain keywords
6. Alter the set K and W and update as K_{NEW} and W_{NEW}

Weight-Fixation:

```
Loop i from 1 to n ; where 'n' is the number of mails in inbox
|  Select mail m_i where m_i ∈ n ∀ i ∈ I+
|    String s[ ] ← k_i from m_i ;  // k_i represents keywords in m_i
|        Form Binary Search Tree (B_t) ;
|  Loop from 1 to n
|  |      Check(m_i,k_j) ∀ 1≤i≤n ∧ k_j ∈ K_NEW
|  |      if match found
|  |  |       Then, W_mi = W_mi + [(f*W_NEW i) + W_mi/10]
|  |      Map m_i to the total weight of m_i;
|  If (W_mi ≥ W_t )
|  |      Then m_i ← spam
|  Else
|  |      m_i ← non-spam
End loop
```

Output: $m_i \leftarrow$ Spam / Ham

Fig. 2: Pseudo-code for UMLA method

If we assume a text mail containing 'n' words, then simple iterative searching method will make a complexity of O(n) for each word in the list. To avoid that amount of complexity for searching every word from the mails, we have divided the task into two sub-tasks:

i. Formation of a Binary Search Tree with the keywords of each mail in alphabetical ascending order.
ii. Searching each keyword from the Binary Search Tree (B_t) and compared with the latest formed keyword list (K_{NEW}).

This searching procedure will have an overall complexity of $O(\log_2 n)$ for searching each keyword after the Binary Search Tree is formed. If we had followed the trivial searching method, then for every word, we had to search the entire text and for every iteration, the complexity would have been O (n). If we had followed the trivial searching method, then for every word, we had to search the entire text and for every iteration, the complexity would have been O(n) as shown by green line in Fig. 3. The X-axis represents number of mails and the Blue line represents the complexity of our searching method.

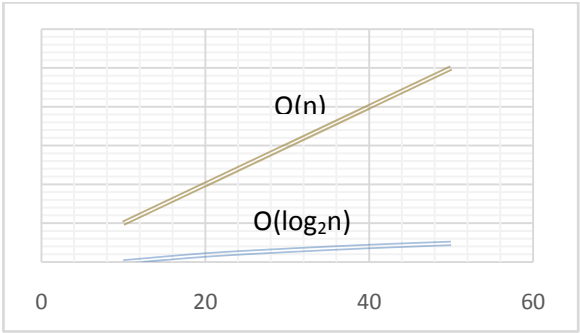


Fig. 3: **Searching time comparison of traditional spam-detection methods Vs. UMLA**

This searching technique is applied in finding the words from the keyword list along with the occurrence of the words and finally calculating the weight of each mail. To calculate the weight of each mail we have proposed three different approaches.

i. Frequency-based weight fixation
ii. Matrix-based weight fixation
iii. Tree-based weight fixation

Basis of Weight Fixation:

We have taken 1000 e-mails from the sample data set and we denote them as $m_1, m_2, m_3, m_4 \ldots m_{1000}$. Out of these 1000 mails there might be some spam mails, which we are about to detect. Three possible techniques of weight fixation we have used in our method depending on context of the mails.

i. Frequency based Weight Fixation:

In this procedure, firstly we follow the searching technique and put the e-mails in a list according to the keywords found along with its frequency of occurrence as shown in Table 1:

Table 1: Frequency of occurrence of the keywords

free(3)	lottery(3)	discount(3)	deal(2)	offer(2)	sex(2)
m_3(3)	m_7(2)	m_2(1)	m_{13}(2)	m_{37}(2)	m_{489}(1)
m_{50}(1)	m_{501}(1)	m_{52}(3)	m_{50}(2)	m_{705}(3)	m_{802}(3)
m_{88}(2)		m_{145}(2)		m_{812}(1)	

From the keyword list, if we multiply the occurrences of each keyword with their respective weights, we can calculate the weightage of each mail and determine whether it is spam or not. The maximum weight allowable is initially fixed as 10, but this may increase based on user's behavior as discussed earlier.

Now, there may situations where a mail has got more than one spam keywords in its content, so to deal with this and to model such situation appropriately we have used the concept of cumulative weight.

If more than one word from the list is found in a mail then the overall weight of each mail is added with a fraction of $1/10^{th}$ of the net weight. Thus, the Cumulative Weighted Sum (CWS) is set as (1/10) based on experimental results on the sample data set.

Let, the weight of a mail 'm_i' be W_{mi} and weight of a particular keyword be 'w' and its frequency in m_i be 'f' then:

$$W_{mi} = \begin{cases} W_{mi} + [(w * f)] & \text{if } (f<2) \\ W_{mi} + [(w * f) + (W_{mi}/10)] & \text{if } (f \geq 2) \end{cases} \quad \ldots\ldots\ldots (1)$$

Table 2 represents some sample mails with their respective cumulative weighted sum values calculated from equation (1).

Table 2: Final CWS values for sample mails with decision

Individual Mail 's	CWS Value	Decision
m_1	0	Ham
m_2	3	Ham
m_3	10	Spam
m_4	7	Ham
m_5	0	Ham
…	…	…
…	…	…
m_{1000}	12	Spam

ii. Matrix based weight fixation:

In this procedure, we have done the weight fixation with the help of matrices. We have used two matrices M_1 and M_2 with 'n' columns, where we assume that the number of keywords in the list is 'n' and each column in the matrix is represented by a keyword from the list of updated keywords (K_{NEW}). The corresponding mail number is assigned to respective column of 1^{st} matrix if that particular keyword is present in that mail.

Let us assume a mail; 'm_i' has a particular keyword, which is represented in the 'j^{th}' column of the matrix, then m_i will be placed in the 'j^{th}' column of the 1^{st} matrix M_1 as shown below.

$$M_1 = \begin{bmatrix} Free & Lottery & Discount & Deal & Offer & Shop & Sex & Order \\ m_3 & m_7 & m_2 & m_{13} & m_{37} & m_{124} & m_{489} & m_{58} \\ m_{50} & m_{501} & m_{52} & m_{50} & m_{705} & m_{288} & m_{802} & m_{95} \\ m_{88} & & m_{145} & & m_{812} & & & \end{bmatrix}$$

The 2^{nd} matrix M_2 contains the frequency of each keyword in the respective position of each element of the 1^{st} matrix. For example, if mail m_3 is in the position [row, column] = [1,1] of the 1^{st} matrix, then [1,1] of the 2^{nd} matrix will signify the frequency of that keyword, represented by the column 1, in m_3.

$$M_2 = \begin{bmatrix} Free & Lottery & Discount & Deal & Offer & Shop & Sex & Order \\ 3 & 2 & 1 & 2 & 2 & 2 & 1 & 4 \\ 1 & 1 & 3 & 2 & 3 & 2 & 3 & 2 \\ 2 & – & 2 & – & 1 & – & – & – \end{bmatrix}$$

If we multiply the element of M_2 having value greater than 0, with the weight of the keyword that is represented by the respective column of the M_1 and repeat this process for every value of the 2^{nd} matrix then we shall get the weightage of each mail and subsequently identify which of the mail is spam or non-spam. This measure depends on whether the weightage crosses the designated threshold value or not. If a mail has more than one keyword, then the Cumulative weighted sum formula will be applied to fix the weight as stated in the first case.

Let, 'n' denotes the number of keywords in the list, two 2-D arrays, mail[ ][ ] and freq[ ][ ] contain the number of mails and the frequency of each keyword in that mail respectively. Array list[ ] contain the weight of the keywords in the same order in which they are arranged and another vector weight[ ] contains the weight of each mail. The pseudo-code of the matrix based weight fixation is given in Fig.4

```
for row = 0 to n
    for col = 0 to n
      do
        if ( mail[row][col] ≠ 0 ) then
          if ( freq[row][col] < 2 ) then
            weight[mail[row][col]] ← freq[row][col]*list[col];
           if(weight[mail[row][col]] > 10) then
              move_to_spam(mail[row][col]);
          else if ( freq[row][col] ≥ 2 ) then
            weight[mail[row][col]] ← weight[mail[row][col]]
                                      + (list[col]* freq[row][col])
                                    +(weight[mail[row][col]] ÷ 10 );
          if(weight[mail[row][col]] > 10) then
              move_to_spam(mail[row][col]);
    done
end loop
```

Fig. 4: Pseudo-code for Matrix based weight Fixation

iii. Tree based weight fixation

In this procedure, we have used the structure of a bipartite tree as shown in Fig. 5. Square and circular shapes represent the two set of nodes where square node signifies the list of keywords generated by the User based Machine Learning Algorithm (UMLA) and the circular nodes signify list of mails to be considered.

Followed by the searching algorithm, we connect the mails with the keywords in a manner such that if any mail (say m_i)

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 6– No.10, April 2017 – www.caeaccess.org

contains a keyword (say k_i) which belongs to the keyword list, then there will exist an edge from the keyword (k_i) node to the mail (m_i) node and the weightage of the edge will be the frequency of the keyword in the mail.

Let us assume that mail m_i contains a keyword k_i and the occurrence of that keyword in the mail is 'f', then there will be an edge from node k_i to node m_i and the weight of the edge will be 'f'. So, mathematically the Weight of the mail m_i following the method of Cumulative Weighted Sum (discussed previously) will be as:

$$W_{mi} = \begin{cases} W_{mi} + [(w * f)] & \text{if } (f<2) \\ W_{mi} + [(w * f) + (W_{mi}/10)] & \text{if } (f>2) \end{cases}$$

From Fig. 5, we can see that there is a connection from node m_{17} to keyword "Lottery" and the weightage of the connection is '2'. This means that mail m_{17} has the keyword "Lottery" with a frequency of 2. Following the aforesaid formula, the weight calculation for a sample mail m_{17} is shown below

$$W_{m17} = 0 + [(2*3) + 0] = 6$$

Similarly, we can determine the weightage of every mail from the tree based model.

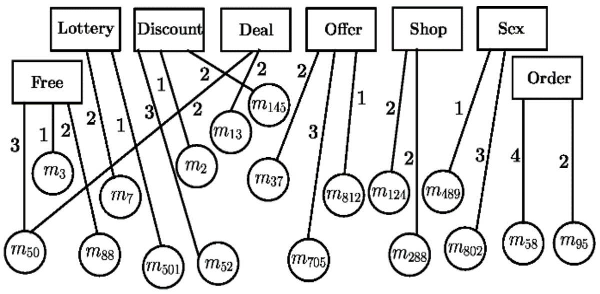


Fig. 5 Bipartite tree showing the mapping of mails to keywords

7. SAMPLE STUDY

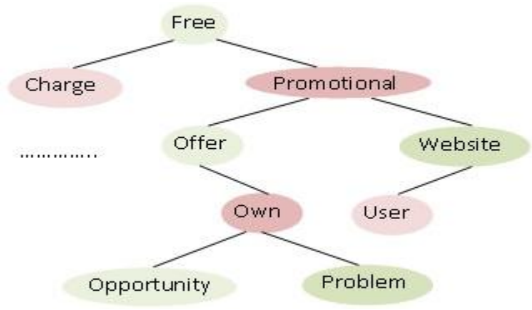
The experiments in this work are done based on the data set named LingSpam [16]. For example, one sample mail taken from the above data set is given below:

"*Subject: free promotional offer*

*' ' own 100 % free web site site : http : / / 138 . 27 . 44 . 5 .cearth . . ca / users / freewebsites / * * * charge * * * * * * commitment * * * * * * problem * * * opportunity seekers internet marketers small large site is . earn prosperous income giving away free web sites . . . already web site ? site linked thousands web sites ? amazing site . . . http : / / 138 . 27 . 44 . 5 . cearth . . ca / users / freewebsites / * * * charge * * * * * * commitment * * * * * * problem * * * is truly going site century ! * please excuse intrusion . one free offer mailing * * * * * * * * * * * ***

Taking the sample mail shown above, the procedure of weight fixation is demonstrated. We first form a Binary Search Tree taking the keywords of the sample mail and then applied binary search. Secondly, we will then search each keyword from the keyword list and note the frequency of each keyword in the mail.

Fig. 6 Binary Search Tree made with the keywords from



the sample mail

The result shows the occurrence of keyword 'free' for 5-times, 'offer' for 2-times. Now, we apply the Cumulative weighted sum formula to calculate the weight as follows.

$$W_1=0+[(3*3)+(0/10)] = 9 \quad (2)$$

$$W_2=9+[(2*2)+(9/10)]=13.9 \quad (3)$$

So, the weight comes to be 13.9, which on initial condition is definitely a spam. But, if analyze the user's inbox and check other mails, then we can surely ascertain the proper score to mark a mail as spam. Thus, we did experiments on 1000 mails found out the result for each mail and tallied the level of accuracy.

8. EXPERIMENTAL RESULTS

The available spam mail filtering procedures are server-side in general, i.e. they have a centralized approach or algorithm for every user and as a result of this, in most of the cases, the effectiveness of filtering spam somehow decreases. To check the effectiveness of the algorithm we have used a sample of 1000 mails from LingSpam dataset [16]. Out of the 1000 mails there were 255 spam mails and the algorithm based on the matrix-based weight fixation in the 2nd technique detected 260 spam mails of which 240 were spam and 20 were non-spam mails and 20 spam mails remained undetected.

So, based on the dataset chosen, the true-positive rate is 88%, false-negative is 8%, false-positive is 8% and true-negative is 97.3%. So, the accuracy of the algorithm based on matrix technique, the total accuracy is 95%. In addition, if we compare the accuracy level of this algorithm with the commercially available spam mail filtering software's, then also it is noticeable that proposed UMLA algorithm is more efficient. (The testing is done based on same dataset)

So, this algorithm proves to be efficient if compared with the efficiency of some commercially available spam filtering software applications. Using same dataset if we compare the accuracy level of other algorithms on which related works are done like -Naive-Bayes method, Logistic Model Tree (LMT) and J48 decision tree classifiers, then also we have achieved much improved performance for our algorithm as shown in Fig. 7 by bar chart diagram compared with other spam detection mechanisms [7].

From the experimental results shown in Table 3 and in Fig. 7, we can clearly suggest that the proposed algorithm is much more efficient than the commonly used detection methods and other commercially available software applications.

Table 3: Comparative study with commercially available Text-spam filters

	Mail-Washer Pro. [17]	Spamihilator [17]	Our Algorithm
True Positive Rate (TPR)	84%	82%	88%
False Negative Rate (FNR)	14%	10%	8%
False Positive Rate (FPR)	10%	8%	8%
True Negative Rate (TNR)	91%	93%	97.3%

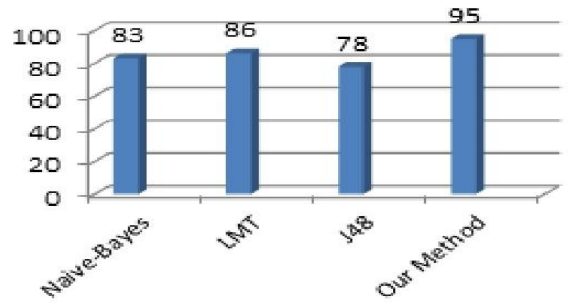


Fig 7: Comparative Accuracy Level with other popular text-spam detection methods

9. CONCLUSION

In this present work, we have designed an algorithm based on Weight fixation, Cumulative weighted sum, and Browser Intelligence System (BIS), which will benefit users to get rid of spam mails and achieve a better mailing experience. The weight fixation procedure consists of three alternative techniques- First, the process of indexing, second, the arrangement in matrix format and thirdly, the arrangement in tree order. Out of these three procedures, the matrix format is most accurate and efficient for general-purpose mails. From the analysis of result, it shows that UMLA, if implemented in a client-side application can detect the issue of spam mail at an accuracy level of 95%.

The proposed UMLA method for spam mail filtering is based on client-side application, which is of its first kind and thus the effectiveness of such filtering increases manifold. Firstly, we can get much more specific, accurate and customized results for every user. As the algorithm involves an analysis of every user's inbox and browsing history, it knows user's choice, preferences, browsing pattern, contacts, and the type of mails the user is interested. This increases the quality and efficiency of the algorithm and provides a much better user experience compared to other spam detection methods. Secondly, as the application based on the algorithm is client-side, the issues regarding security is also solved as no third party except the client and the server can peep through the content of a mail.

Therefore, we can conclude our approach can be used to develop an effective spam mail filter but only after properly justifying the scope of improvement in terms of false positive rate. User's criteria oriented image-spam detection can also be targeted separately based on machine learning techniques as future scope of this work.

10. REFERENCES

[1]. Christina V, Karpagavalli S, Suganya G, "A Study on Email Spam Filtering Techniques", International Journal of Computer Applications (0975 – 8887) Volume 12– No.1, December 2010

[2]. Saadat Nazirova, "Survey on Spam Filtering Technique", Communications and Network, 2011, 3, 153-160 doi:10.4236/cn.2011.33019 Published Online August 2011

[3]. Cormack G (2008) Email spam filtering: a systematic review. Found Trends InfRetr 1(4):335–455

[4]. V. Christina et al. Email Spam Filtering using Supervised Machine Learning Techniques. International Journal on Computer Science and Engineering (IJCSE) Vol. 02, No. 09, 2010, 3126-3129

[5]. S.Dhanaraj, Dr. V. Karthikeyani, "A Study on E-mail Image Spam Filtering Techniques", Pattern Recognition, Informatics and Mobile Engineering (PRIME) February 21-22

[6]. Lamia Mohammed Ketari, Munesh Chandra, Mohammadi Akheela Khanum, "A Study of Image Spam Filtering Techniques", 2012 Fourth International Conference on Computational Intelligence and Communication Networks

[7]. Sarit Chakraborty, Bikramadittya Mondal, "Spam Mail Filtering Technique using Different Decision Tree Classifiers through Data Mining Approach - A Comparative Performance Analysis", International Journal of Computer Applications, 47(16):26-31, June 2012 (ISSN: 0975 – 888)

[8]. H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[9] S. Ruggieri, "Efficient c4. 5 [classification algorithm]," *IEEE transactions on knowledge and data engineering*, vol. 14, no. 2, pp. 438–444, 2002

[10]. W. Feng, J. Sun, L. Zhang, C. Cao and Q. Yang, "A support vector machine based naive Bayes algorithm for spam filtering," *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, Las Vegas, NV, 2016, pp. 1-8

[11]. Tarek M Mahmoud, Alaa Ismail El Nashar, Tarek Abd-El-Hafeez and Marwa Khairy, "An Efficient Three-phase Email Spam Filtering Technique", British Journal of Mathematics & Computer Science 4(9), 2014.

[12]. Pingchuan Liu and Teng-Sheng Moh ,"Content Based Spam E-mail Filtering", 2016 International Conference on Collaboration Technologies and Systems

[13]. Ahmed Khorsi, "An Overview of Content-based Spam Filtering Techniques", Informatica, vol. 31, no. 3, October 2007, pp 269-277.

[14]. J. W. Yoon, H. Kim, and J. H. Huh, "Hybrid spam filtering for mobile communication," *computers & security*, vol. 29, no. 4, pp. 446–459, 2010.

[15]. B. Sch Ikopf, S. Mika, C. Burges *et al.*, "Input space versus feature space in kernel-based method," *IEEE Trans Neural Networks*, pp. 1000–1017

[16]. Ling-Spam data set has been taken from - www.csmining.org

[17]. The Spam-mail filters used for testing in Table 3 are taken from www.fireturst.com, www.spamihilator.com